(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2001－142692
（P2001－142692A）
(43)公開日　平成13年５月25日(2001.5.25)

(51)Int.Cl.⁷		識別記号	FI		テーマコード(参考)
Ｇ０６Ｆ	9/30	３１０	Ｇ０６Ｆ	9/30	３１０Ｅ
	9/32	３２０		9/32	３２０Ａ

審査請求　未請求　請求項の数26　ＯＬ　（全 22 頁）

(21)出願番号　特願2000－278071(P2000－278071)
(22)出願日　平成12年９月13日(2000.9.13)
(31)優先権主張番号　０９／４１１１４０
(32)優先日　平成11年10月１日(1999.10.1)
(33)優先権主張国　米国（ＵＳ）

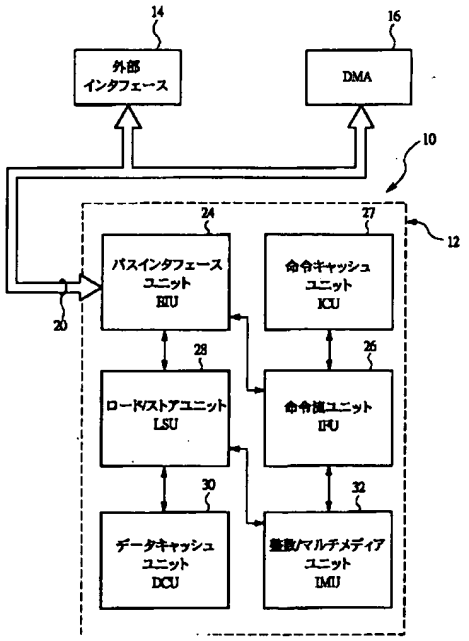
(71)出願人　000005108
株式会社日立製作所
東京都千代田区神田駿河台四丁目６番地
(72)発明者　シバラム・クリシュナン
アメリカ合衆国、カリフォルニア州
94024、ロスアルトス、ウエストプルック・アベニュー　1723
(74)代理人　100080001
弁理士　筒井　大和

最終頁に続く

(54)【発明の名称】　２つの異なる固定長命令セットを実行するマイクロプロセッサ、マイクロコンピュータおよび命令実行方法

(57)【要約】
【課題】　３２ビット固定長命令セットアーキテクチャを実行するように構成され、１６ビット固定長命令セットアーキテクチャと後方互換性を持つプロセッサエレメントを提供すること。
【解決手段】３２ビット固定長命令セットアーキテクチャを実行するように構成されたプロセッサエレメント１２は、各１６ビット命令を１つ又は複数の３２ビット命令のシーケンスに変換することによって１６ビット固定長命令セットアーキテクチャと後方互換性がある。１６ビット命令実行と３２ビット命令実行との間のスイッチングは目標命令が１６ビット命令または３２ビット命令のどちらであるかを識別するために分岐の目標アドレスの最下位ビット位置を用いる分岐命令によって達成される。

図 1



1

【特許請求の範囲】

【請求項1】　MおよびNが整数であってMがNより小さい場合に、Mビット命令セット又はNビット命令セットから操作可能なプロセッサであって、
それぞれNビット命令およびMビット命令を有する少なくとも第1および第2命令ストリームを記憶するメモリユニットと、
Nビット命令を実行するために実行信号を受け取るように操作可能な実行ユニットと、
実行信号を生成するために前記メモリユニットから前記第1および前記第2命令ストリームを受け取って復号するように前記メモリユニットおよび前記実行ユニットに結合された復号ユニットであって、前記復号ユニットが、前記復号ユニットによる復号を実施するために、各々の前記Mビット命令を受け取って前記Mビット命令の第1グループの各々を前記Nビット命令の対応する1つに変換し、前記Mビット命令の第2グループの各々を2つ以上のNビット命令へ変換する変換ユニットを含む復号ユニットと、
からなるプロセッサ。

【請求項2】　前記Mビット及びNビット命令の各々が、Mビット命令のメモリアドレスを識別するための第1状態およびNビット命令のメモリアドレスを識別するための第2状態に設定された少なくとも1つのビット位置を有するメモリアドレスによって識別される前記メモリユニット内の場所に記憶される請求項1に記載のプロセッサ。

【請求項3】　前記Nビット命令ストリームが少なくとも1つのNビット分岐命令を含み、実行中のNビット命令から実行中のMビット命令へスイッチするために前記Nビット分岐命令を実行するように前記デコードユニットが作動する請求項2に記載のプロセッサ。

【請求項4】　Nが2Mに等しい請求項1に記載のプロセッサ。

【請求項5】　Mが16であり、Nが32である請求項1に記載のプロセッサ。

【請求項6】　1つのビット位置が前記メモリアドレスの最下位ビットである請求項2に記載のプロセッサ。

【請求項7】　プロセッサユニットであって、
複数の命令を記憶するメモリであって、MとNが整数でMがNより小さいMビット命令およびNビット命令を含み、各命令がメモリアドレスによって識別された記憶場所に記憶され、各メモリアドレスが前記Mビット命令に関する第1状態および前記Nビット命令に関する第2状態へ設定された1つのビット位置を備えてなるメモリと、
検索された命令の実行を制御するためのメモリから前記命令を検索するための命令流制御ユニットであって、前記命令流制御ユニットが、2つ以上の前記Nビット命令のシーケンスに変換するために、前記Mビット命令の幾つかを受け取るように操作可能な変換ユニットを含む命令流制御ユニットと、
からなるプロセッサユニット。

2

【請求項8】　マイクロプロセッサであって、
MとNが整数であってMがNよりも小さい場合に複数のMビット命令およびNビット命令を含むメモリエレメントと、
前記Mビット命令またはNビット命令の選定された幾つかを検索するために前記メモリエレメントに結合された命令フェッチユニットとを有し、前記命令フェッチユニットは、
前記メモリエレメントからフェッチされた前記Mビット命令の各々を1つ又は複数のNビット命令のシーケンスに変換するトランスレータユニットと、
前記メモリエレメントからフェッチされたNビット命令の各々及び前記トランスレータユニットからNビット命令の各々を受け取って、この種のNビット命令を復号するために前記メモリエレメント及び前記トランスレータユニットへ結合される復号ユニットとを含む、
マイクロプロセッサ。

【請求項9】　目標アドレスを保持するための少なくとも1つの目標レジスタを含み、複数の前記Nビット命令が、前記マイクロプロセッサによって実行される時に目標レジスタに目標アドレスをロードする準備目標命令を含む請求項8に記載のマイクロプロセッサ。

【請求項10】　前記複数のNビット命令が、それぞれ変換または復号するために1つのMビット目標命令または1つのNビット目標命令を前記メモリエレメントからフェッチさせるように分岐するために前記目標レジスタ内の前記目標アドレスを使用するように作動するＢＬＩＮＫ分岐命令を含む請求項9に記載のマイクロプロセッサ。

【請求項11】　前記ＢＬＩＮＫ分岐命令が無条件分岐命令である請求項10に記載のマイクロプロセッサ。

【請求項12】　前記目標アドレスがMビット目標命令を識別するために第1状態に設定された1つのビット位置を含む請求項9に記載のマイクロプロセッサ。

【請求項13】　前記ビット位置がNビット目標命令を識別するために第2状態に設定される請求項12に記載のマイクロプロセッサ。

【請求項14】　MとNが整数であってNがMより大きいMビット命令およびNビット命令を記憶するメモリと、
受け取った各Mビット命令を1つ又は複数のNビット命令のシーケンスに変換するためのMビット命令を受け取るために前記メモリに結合されたトランスレータと、
前記Nビット命令を受け取って復号するために前記メモリおよび前記デコーダに結合されたデコーダとを含む、
単一チップ上に形成されたマイクロコンピュータ。

【請求項15】　Mが16であり、Nが32である請求

項１４に記載のマイクロコンピュータ。

【請求項１６】　前記Ｎビット命令が目標命令の分岐アドレスを表示するデータを含むＮビット分岐命令を含み、前記目標命令がＭビット命令である時に第１状態に設定された１つのビット位置を前記分岐アドレスが有する請求項１４に記載にマイクロコンピュータ。

【請求項１７】　ＭとＮが整数であってＭがＮより小さい場合に単一チップ上に形成されたマイクロコンピュータによってＭビット命令およびＮビット命令を実行する方法であって、

前記Ｍビット命令および前記Ｎビット命令をメモリに記憶するステップと、

前記Ｎビット命令の幾つかを順次復号するように第１モードにおいて操作するステップと、

前記Ｍビット命令の幾つかを１つ又は複数のＮビット命令のシーケンスに順次変換するように第２モードにおいて操作し、次に、前記Ｎビット命令を復号するステップと、

を含む命令実行方法。

【請求項１８】　前記Ｎビット命令が１つのＮビット分岐命令を含み、前記第１モードにおいて操作するステップが、操作の前記第１状態から第２状態へスイッチするために第１状態に設定された最下位ビットを含むメモリアドレスを有する１つのＭビット命令へ分岐するように前記Ｎビット分岐命令を復号するステップを含む請求項１７に記載の方法。

【請求項１９】　Ｎビット分岐命令を含むＮビット命令、及び、Ｍビット分岐命令を含むＭビット命令を実行するように構成されたマイクロコンピュータにおいて、ＭとＮが整数であってＮがＭより大きい場合に、Ｍビット命令を実行する方法であって、

１つ又は複数のＮビット命令のシーケンスでＭビット分岐命令の各々をエミュレートするステップと、

分岐アドレスを提供する準備分岐命令で、その後で、前記分岐アドレスを使用するＮビッ分岐命令でＭビット分岐命令をエミュレートするステップと、

を含む命令実行方法。

【請求項２０】　少なくとも１つの目標アドレスレジスタを提供するステップを含み、前記Ｍビット分岐命令をエミュレートするステップは、前記目標アドレスレジスタに目標アドレスをロードする準備分岐命令を含む請求項１９に記載の方法。

【請求項２１】　前記Ｍビット分岐命令をエミュレートするステップが、前記Ｎビット分岐命令が前記目標アドレスを読み取るステップを含む請求項２０に記載の方法。

【請求項２２】　各々が１つ又は複数の前記Ｎビット命令に応答してデータを記憶するための複数の汎用レジスタを含む請求項１４に記載のマイクロコンピュータ。

【請求項２３】　複数の前記汎用レジスタの各々が２Ｎビット位置を含む請求項２２に記載のマイクロコンピュータ。

【請求項２４】　１つ又は複数の前記Ｎビット命令の中の所定個数のＮビット命令が、複数の前記汎用レジスタの中の選定されたレジスタの低位ビット位置にデータをロードする請求項２２に記載のマイクロコンピュータ。

【請求項２５】　１つ又は複数の前記Ｎビット命令の中の選定された第１のＮビット命令が、複数の前記汎用レジスタの中の選定された１つの低位ビット位置にデータをロードし、１つ又は複数の前記Ｎビット命令の中の選定された第２のＮビット命令が、複数の前記汎用レジスタの高位ビット位置にデータをロードする請求項２２に記載のマイクロコンピュータ。

【請求項２６】　１つ又は複数のＮビット命令の中の所定個数のＮビット命令が、複数の前記汎用レジスタの中の選定されたレジスタの高位ビット位置において複製されたデータの最上位ビットのエキステンションと共に複数の前記汎用レジスタの中の選定されたレジスタの低位ビット位置にデータをロードする請求項２２に記載のマイクロコンピュータ。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は一般にマイクロプロセッサ／マイクロコントローラアーキテクチャに関し、詳細には、第２の更に小さい固定した命令への後方互換性を備えた第１の固定長命令セットを実行するように構成されたアーキテクチャに関する。

【０００２】

【従来の技術】電子産業におけるミニアチュア化およびパッケージングの分野における最近の進歩は様々な「埋め込まれた」製品の設計に関して機会を提供している。埋め込まれた製品は一般に小さくかつ手持ち式であり、制御機能用のマイクロコントローラ又はマイクロプロセッサを含むように構成される。埋め込まれた製品の例には、例えばセル電話、ポケットベル（登録商標）、及び、個人用デジタル補助装置（ＰＤＡ）にような手持ちビジネス用、消費者用、および、工業用デバイスが含まれる。

【０００３】埋込み設計すなわちアーキテクチャを成功させるには、例えば、埋め込まれる部品のサイズ及び電力消費のような或る種の必要条件を考慮しなければならない。この理由に因り、埋め込まれた製品用の或る種のマイクロコントローラ及びマイクロプロセッサは、命令の比較的小さいセットの迅速かつ効率的な処理に焦点を絞った縮小命令セット計算（ＲＩＳＣ）アーキテクチャを組み入れるように設計されている。ただし、初期のＲＩＳＣ設計は３２ビットの固定長命令セットを使用した。処理エレメントを更に最小限化するために、例えば１６ビットのような小さな固定サイズを使用する設計が開発され、コンパクトなコードの使用を可能にし、命令

メモリのサイズを縮小した。小さくコンパクトなコードと結合したＲＩＳＣアーキテクチャは埋め込まれた製品の設計が更に簡潔、小型、電力重視的であることを可能にする。この種の１６ビットアーキテクチャの一例は米国特許第５，６８２，５４５号に開示されている。

【０００４】

【発明が解決しようとする課題】しかし、１６ビット命令セットによって提供可能であるよりも更に大きな計算能力および融通性の必要性が存在し、特にグラフィックスに対する能力が要求される時にそのような必要性が大きくなる。この必要性を満たすために、３２ビット命令セットアーキテクチャが利用可能にされつつある。ただし、この種の３２ビット命令セットアーキテクチャを用いると、一層大きい３２ビット命令を記憶するための一層大きいメモリサイズが必要とされる。更に大きいメモリサイズは、埋め込まれた製品の設計の成功に逆行する傾向のある更に高い電力消費およびより多くの空間に関する必要条件の必要性を伴う。

【０００５】更に、現在の３２ビット命令セットアーキテクチャは、該当する場合には、初期に開発された１６ビットコードに対する後方互換性を殆ど提供しない。その結果として、実質的なソフトウェア投資は失われる。従って、従来の小さい方のコードを使用するアプリケーションは廃棄するか、又は、３２ビット命令に対してコンパイルし直さなければならない。

【０００６】従って、サイズ及び電力消費拘束条件に殆ど影響を及ぼすことがないと同時に、初期の命令セットアーキテクチャへの後方互換性を提供する３２ビット命令アーキテクチャを提供する必要性のあることが分かるはずである。

【０００７】

【課題を解決するための手段】本発明は、概して、大きい方の固定長命令セットアーキテクチャ又は初期に設計された小さい方の固定長命令セットアーキテクチャのどちらでも実行するように構成され、これによって小さい方の命令セットへの後方互換性を提供する例えばマイクロプロセッサ又はマイクロコントローラのようなプロセッサエレメントに向けられる。小さい方の命令セットの実行は、主として、大きい方の命令の１つ又は複数のシーケンスを用いて小さい方の各命令をエミュレートすることによって達成される。更に、小さい方の命令セットアーキテクチャの資源（例えば、レジスタ、状態ビット、および他の状態）は大きい方の命令セット環境の資源にマップされる。

【０００８】本発明の一実施の形態において、大きい方の命令セットアーキテクチャは３２ビット固定長命令を使用し、小さい方の命令セットは１６ビット固定長命令を使用する。ただし、当技術分野における当業者は分かるはずであるように、２つの異なる命令セットは任意の長さであっても差し支えない。１６ビット命令の第１グループは、それぞれ、１つの単一３２ビット命令シーケンスによってエミュレートされる。１６ビット命令の第２グループは、それぞれ、２つ以上の３２ビット命令のシーケンスによってエミュレートされる。当該分岐の目標が一方の命令セット（例えば１６ビット）のメンバーであることを識別するために所定の状態、又は、当該目標がもう一方の命令セット（３２ビット）のメンバーであることを識別するために反対の状態にセットされた１つのビット位置（好ましい実施形態における最下位ビット（ＬＳＢ））を持つ目標アドレスを使用する分岐（分岐）命令により、実行のモード間のスイッチングが達成される。

【０００９】特定の１６ビット命令セットアーキテクチャは分岐命令に関するいわゆる「遅延スロット」を含む。遅延スロットは分岐命令に密接して後続する命令であり、分岐命令の或る態様の準備期間中であって、当該分岐（分岐）が実施される以前に実行される（分岐命令がそのように指示する場合）。この様式において、分岐に関するペナルティが軽減される。遅延スロット命令を伴う１６ビット分岐命令をエミュレートすることは、目標レジスタをロードする分岐命令に先行する準備分岐（ＰＴ）命令を用いて達成される。次に、分岐命令は、分岐用目標レジスタの内容を使用する。ただし、遅延スロット必要条件付き１６ビット命令をエミュレートする時は、分岐は実行されるが、１６ビット遅延スロット命令のエミュレーション及び実行が完了するまで、目標命令（分岐が実施される場合）は一時停止状態に保持される。

【００１０】３２ビットＰＴ命令は、分岐目標プロセッサエレメントの通知を分岐命令から分離することによって、３２ビット命令セット環境において低ペナルティ分岐を提供するように作動する制御流れ機構の一部を形成する。これは、プロセッサハードウェアが先行する多くのサイクルに亙る分岐に気づかされることを可能にし、現行命令シーケンスから目標シーケンスへの円滑な遷移を可能にする。更に、それは、分岐ペナルティを最小限化するために１６ビット命令セットアーキテクチャにおける遅延スロット技法使用の必要性をなくする。

【００１１】本発明の特徴は、１６ビット命令または３２ビット命令のどちらでも使用することにより、各々の長さが６４ビットの幾つかの汎用ないし多目的レジスタを提供する。ただし、汎用レジスタは、低位の３２ビットが使用される場合に限り、１６ビット命令によって書込みまたはロードされる。更に、符号ビットの自動エキステンションは、大抵の１６ビット命令が汎用レジスタをロードする時に実施される。即ち、６４ビット汎用レジスタの低位ビット位置に置かれた３２ビット量の最上位ビットが、当該レジスタの３２の高位ビット全てにコピーされる。３２ビット命令セットアーキテクチャは、１６ビットと３２ビット環境の間に互換性を提供するた

めにこのプロトコルを使用するように構成された命令を含む。

【0012】同様に、64ビット状態レジスタは16ビット命令セット及び32ビット命令セットの両方用に装備されている。状態レジスタの所定ビット位置は、16ビット命令セットからマップされる状態用に予約される。他の16ビット状態は汎用レジスタの所定ビット位置にマップされる。16ビット命令セット状態のこのマッピングは、タスク切り替えに必要な全ての文脈をセーブするために、個別環境（16ビット、32ビット）を可能にし、32ビット命令を用いた16ビット命令のエミュレーションを容易にする。

【0013】多数の利点が本発明によって達成される。16ビットコード及び32ビットコードの両方を実行する能力は、プロセッサが普通のタスクのためにコンパクトな16ビットコードを使用することを可能にする。次に、これは、両メモリスペースの節減および当該節減を伴った他の利点（例えば、更に小さいメモリ、節減された電力消費等）を可能にする。更に複雑なタスクが必要とされるときは、32ビットコードが使用できる。

【0014】更に、初期に設計された16ビット命令セットアーキテクチャを実行する能力は、当該初期設計においてなされた投資の保持を可能にする互換性を提供する。

【0015】PT命令は、分岐に関する事前通知を提供することにより、分岐命令の遂行におけるより多くの融通性を可能にする。

【0016】本発明のこれらの及び他の利点および特徴は、添付図面と共になされる以下の詳細な記述を読むことにより、当該技術分野における当業者にとって明白になるはずである。

【0017】

【発明の実施の形態】本発明は好ましくは既に開発済みの16ビット固定長命令セットアーキテクチャに後方互換性を提供する。この種アーキテクチャの更に完全な記述は「SH7750 Programming Manual」（プログラミングマニュアル）（Rev. 2.0、版権1999年3月4日）（Hitachi Semiconductor (America) Inc., 179 East Tasman Drive, San Jose, CA 95134から入手可能）に記載されている。

【0018】さて、図面、特に図1を参照することとし、ここに本発明の教示に従って作成されたプロセッサエレメント（例えばマイクロコンピュータ）の概略ブロック図を示す。図1に示すように、一般に参照番号10によって識別されるプロセッサシステムは、プロセッサエレメント12、外部インタフェース14、及び、システムバス20によって相互接続された直接メモリアクセス（DMA）ユニット14を含む。外部インタフェース14は、外部メモリに接続するように構成されることが好ましく、他の処理エレメント（例えば周辺デバイス、通信ポート、等）への通信的アクセスをプロセッサエレメント12へ提供しても差し支えない。

【0019】また、図1は、プロセッサユニット12を外部インタフェース14及びDMA16とインターフェイスするバスインタフェースユニット（BIU）24をプロセッサエレメント12が含む状態を示すことによって、プロセッサエレメント12の論理的配分も図示する。外部インタフェース14を介してシステムバス20及び外部メモリ（図示せず）に対するおよびこれらからの全てのリクエストを処理するBIU24は命令流れユニット（IFU）26へ通信的に接続する。IFU26は、命令キャッシュユニット（ICU）27からフェッチする命令を復号し、命令復号および実行パイプラインのフロントエンドとして貢献するように作動する。図に示すように、IFU26は、本発明に従って16ビット命令セットに32ビット命令セットのシーケンスをエミュレートさせる翻訳（ないし変換）論理を含む（以後、16ビット命令セットアーキテクチャを「モードB」と称し、32ビット命令セットアーキテクチャを「モードA」と称することとする）。

【0020】同様に、BIU24は全てのメモリ命令を処理し、データキャッシュユニット（DCU）30の操作を制御するプロセッサエレメント12のロード／ストアユニット（LSU）28へ接続する。整数／マルチメディアユニット（IMU）32は、全ての整数およびマルチメディア命令を処理するためにプロセッサエレメント12に含まれ、プロセッサエレメント12用の主データパスを形成する。

【0021】IFU26は、主要部分において、プロセッサエレメント12のシーケンサとして機能する。その主要機能は、ICU27から命令をフェッチし、それらを復号し、レジスタファイル50（図2）からオペランドを読み取り、復号済み命令およびオペランドを実行ユニット（IMU32及びLSU28）へ送り、実行ユニットから結果を収集し、それらをレジスタファイルへ書き戻す。その上、外部メモリ（図示せず）からの紛失命令で命令キャッシュを満たすことに命令キャッシュが失敗すると、IFU26はメモリリクエストをBIU24に発出する。

【0022】IFUの別の主要タスクはモードB命令のエミュレーションを実施することである。詳細には、特定のモードB命令がモードA命令の1つ又は一連のモードA命令のどちらかによってエミュレートされるように全てのモードB命令が翻訳される。次に、モードA命令は、元のモードA命令意味論へ、極めて小さい変更を伴って実行される。この方法は、モードB命令を実施するために必要な回路および論理が少数の機能論理ブロック内において分離されることを可能にする。これは、将

来、モードＢ命令セットの変化を可能にするか、又は、おそらく更に重要には、モードＢ命令セットを完全に除去することを可能にする利点を持つ。

【００２３】図２はＩＦＵ２６の構成を一層詳細に示すブロック図である。プロセッサエレメント１２内における順序付けはＩＦＵ２６によって行われるので、ＩＦＵは、プロセッサエレメント１２の他の殆ど全てのユニットとインターフェイスされている。ＩＦＵ２６とＢＩＵ２４及びＩＣＵ２７両ユニットとの間のインタフェースは、ＩＣＵ２７への命令のローディングおよび実行に関するＩＣＵ２７からの命令の流れを処理するＩＣＡＣＨＥ（命令キャッシュ）コントロール（ＩＣＣ）４０によって確立される。ＩＣＵ２７とＬＳＵ２８及びＩＭＵ３２の間のインタフェースは、命令、オペランド、結果、ならびに、命令の実行を可能化する全ての制御信号の送信／受信経路を供給する。これらのインタフェースに加えて、同様にＩＦＵ２６は、外部割込みをサンプリング及び調停する外部割込みコントローラ４１から外部割込み信号も受け取る。次に、ＩＦＵ２６は外部割込みと内部例外を調停し、非同期イベントを取り扱う適当なハンドラを作動化する。

【００２４】図２に示すように、ＩＣＣ４０は、アクセスを準備するために、内部的にはフェッチユニット（ＦＥ）４２と、外部的にはＩＣＵ２７と交信する。通常、ＦＥ４２は１つの命令フェッチアドレス、及び、ＩＣＣ４０に「フェッチデマンド」を表示する１組の制御信号を供給する。復帰に際して、ＩＣＣ４０は、最大２つまでの語整列命令ワードをＦＥ４２に送り返す。ＩＣＵ２７がミスする時には、ミスしているキャッシュラインに外部メモリ（図示せず）からロードするために、ＩＣＣ４０はＢＩＵ２４への補充サイクルを開始する。補充は、ＦＥ４２が元のフェッチアドレスを保持している期間中に行われる。その代りに、ＦＥ４２は、命令の返却を要求しない「プリフェッチリクエスト」、あるいは、キャッシュミス（欠落）が起きた時に一切の補充活動を必要としない「フェッチリクエスト」を供給しても差し支えない。

【００２５】ＦＥ４２によってＩＣＵ２７からフェッチされた命令は、命令の命令セットアーキテクチャモード（即ち、モードＢ又はモードＡのどちらか）に従って先ずバッファエリア４２ａに貯蔵される。ただし、最終的に、命令は、復号（ＤＥＣ）ユニット４４への適応のために、２つの命令バッファの１つに移送されるはずである。

【００２６】プロセッサエレメント１２がモードＡ命令を実行しているとき、ＤＥＣ４４は命令を復号し、復号済み命令情報をＦＥ４２、分岐ユニット（ＢＲ）４６、及び、パイプラインコントロール（ＰＰＣ）４８へ、また、外部的にはＩＭＵ３２及びＬＳＵ２８へ送るはずである。情報は、当該命令をそれ以上復号することなく、ＬＳＵ２８及びＩＭＵ３２がデータオペレーションを開始することも可能にする。分岐命令に関しては、部分的に復号された分岐情報は、出来るだけ早期に分岐方向を静的に予測するために、ＢＲ４６を作動可能化する。

【００２７】モードＢ命令が実行中であるときには、全ての命令は追加パイプラインステージ、即ちＤＥＣ４４のモードＢトランスレータ４４ａを通過するはずである。モードＢトランスレータ４４ａは、各モードＢ命令を、１つまたは複数のモードＡエミュレート命令に変換するはずである。次に、モードＡエミュレート命令はＤＥＣ４４のバッファまで移動させられ、ここで、正常モードＡ命令復号および実行が再開始する。一例として、ここに記載される付録Ａは、モードＢ移動および算術命令の各々に関して、モードＢ命令をエミュレートするために用いられるモードＡ命令シーケンスを示す。（既に確認済みのＳＨ７７５０プログラミングマニュアルにおいて見られるように、モードＢ命令セットには、浮動小数点命令を含む更に多くの命令が含まれる。付録Ａは、エミュレーションを示すためにのみ用いられる。）当技術分野における当業者は、エミュレーションシーケンスが特定の命令セットアーキテクチャに依存することを認識するはずである。

【００２８】更に、３２ビットのデータの処理に際して互換性を保証するために、追加モードＡ命令は、モードＢ（３２ビットデータ）命令をエミュレートするためのモードＡ命令セットに含まれる。付録Ｂに示すこれらの追加命令は、当該命令において識別されたソースレジスタの下から３２ビットのみを検索することによって３２ビットデータを処理するように操作する。この操作のあらゆる結果は、当該命令において識別済みの宛先レジスタの下から３２ビットに書き込まれ、書き込まれた量の符号ビット（即ち、最上位ビット）は宛先レジスタの上から３２ビットに拡張されるはずである。

【００２９】モードＡ命令によるモードＢ命令のエミュレーションの一例は、付録Ａに記載されたモードＢ加算（ＡＤＤ）命令によって示される。これは、１つの単一モードＡ命令である１つの加算長（ａｄｄ．ｌ）命令によってエミュレートされるモードＢ命令の１つである。モードＢ命令セットアーキテクチャにおいて、ＡＤＤ命令は、２つの１６汎用レジスタＲｍ、Ｒｎの内容を相互に加算し、汎用レジスタＲｎ内の結果を記憶する。（理解されるであろうように、１６個の汎用レジスタ（R_0－R_{15}）は、６４ビット汎用レジスタ（R_0－R_{15}）５０の低位の３２ビットにマップされる。）このモードＢのＡＤＤ命令のエミュレーションは、汎用レジスタの下から３２ビットのみを用いるモードＡ加算長（ａｄｄ．ｌ）命令を使用する。Ａｄｄ．ｌは、汎用レジスタＲｍの内容を汎用レジスタＲｎの内容に加えるように操作し、汎用レジスタＲｎの下から３２ビットにおける結果を符号ビットの自動エキステンションと共に当該レジス

タの上から３２ビットに記憶する。それによって、同一タスクを実施し、同一３２ビット結果を得るために、モードBのADD命令はモードAのａｄｄ．１命令によってエミュレートされる。（モードA命令は汎用レジスタの６４ビット全体を使用する。レジスタに記入されるべき値が６４ビット全体より少ないならば、モードB命令またはモードA命令のどちらかによって記入される場合であって、大部分が符号なし演算の場合であってもその値の符号は当該レジスタの高い方のビット位置に拡張される。これは、モードBまたはモードA演算の結果が６４ビットの結果を生成するとみなされることを可能にする）。モードB命令セットに関して、上述のSH７７５０Ｐｒｏｇｒａｍｍｉｎｇ　Ｍａｎｕａｌ（プログラミングマニュアル）に記載されているように、加算済みモードA命令は、ここに示す付録Bには記載されていない。

【００３０】一連の２つ以上のモードA命令によるモードB命令のエミュレーションの一例はモードB桁上げあり加算（ADDC）命令により付録Aに示される。レジスタRm、Rnの内容が符号なし数として扱われること、及び、モードB命令セットアーキテクチャの１ビットTレジスタに記憶されている以前の加算によって生成された桁上げが和に含まれることを除けば、ADDC命令はADD命令と同じである。ADDCが桁上げを生成する場合には、ならば、その桁上げは、後続するADDC命令による使用のため又は他の演算のためにモードB環境において１ビットTレジスタに記憶される。これは、一連のモードA命令によるエミュレーションを必要とする。（レジスタに関しては図２のレジスタファイル５０に含まれる６４ビット汎用レジスタを参照されたい。）理解可能であるように、ADDC命令は一連のモードA命令６個によってエミュレートされる。

【００３１】１．モードA加算符号なし長（ａｄｄｚ．１）命令は汎用レジスタR_mの下から３２ビットを汎用レジスタR_{63}（定数「０」）に加え、その結果を、当該結果の上から３２ビットに拡張されたゼロと共に、スクラッチパッドレジスタとして用いられる汎用レジスタR_{32}へ戻す。

【００３２】２．次に、ａｄｄｚ．１は、汎用レジスタRnの下から３２ビットを汎用レジスタR_{63}に加え、その結果を、当該結果の上から３２ビットに記入されたゼロと共に汎用レジスタR_nへ戻す。

【００３３】３．次に、モードA加算命令はR_nとR_{32}との内容を相互に加算し（両者共、下から３２ビット位置に３２ビット量および上から３２ビット位置にゼロを有する）、その結果をレジスタR_nに記憶する。

【００３４】４．モードA加算命令は、現在レジスタR_n内に在る結果を、より早く生成されて、汎用レジスタR_{25}のＬＳＢに置かれている桁上げに加算し、その結果をレジスタR_nへ戻す。

【００３５】ステップ４の結果は１ビットTレジスタに設定されている桁上げを生成することがあり得るので、モードB環境において、Tレジスタがマップされているレジスタ（汎用レジスタR_{25}のＬＳＢ）は、エミュレーションの残りのステップに際して、桁上げと共にロードされる。

【００３６】５．R_nに保持された値は、加算によって生じたあらゆる桁上げを当該値のＬＳＢへ移動させるために、右方へ３２ビット位置シフトされ、その結果はR_{25}に書き込まれる。

【００３７】６．最後に、レジスタR_nの内容は符号拡張された値ではないので、モードA即時加算命令は内容をゼロに加え、符号拡張された結果をR_nに戻す。

【００３８】同様に、当該命令によって用いられるレジスタに記憶済みの値に応じて単一モードA命令または一連のモードA命令２個にエミュレートされたモードB命令がある。この二重特性エミュレーションの例は３移動（Ｍｏｖｅ）データ命令であり、この場合のソースオペランドはメモリ（ソースが@R_mである場合にはMOV．B、MOV．W、MOV．L）である。モードB環境において、これらの命令は、汎用レジスタRmの内容によって指定された記憶場所においてメモリ内データを検索し、それをレジスタR_nの内容に加え、その結果をレジスタR_nに戻す。次に、mがnに等しくないならば（即ち、データは、当該メモリアドレスを保持するレジスタ以外の全ての他のレジスタに移動中である場合）、レジスタR_mの内容はインクリメントされる。付録Aに見られるように、データが、当該データのメモリアドレスを保持する汎用レジスタへメモリから移される場合には、ただ１つの命令が用いられる。そうではなくて、データが他の場所に移動中であれば、当該メモリアドレスは第２の命令によってインクリメントされる。

【００３９】図２へ戻って、BR４６は全ての分岐に関する命令を扱う。BR４６は復号済み分岐命令をDEC４４から受け取り、分岐条件および目標アドレスが既知であるかどうかを決定し、分岐を解決／予測するために進行する。分岐条件が未知であれば、BR４６は分岐条件を静的に予測する。次に、予測された命令はフェッチされて、復号される。場合によっては、予測済み命令がフェッチされ、分岐条件が解決される前に復号されることもあり得る。これが起きると、予測が正しいことをBR４６が確認するまで、予測済み命令は復号ステージに保持される。

【００４０】BR４６は、８個の目標アドレスレジスタ４６ａ、ならびに、状態レジスタ（SR）４６ｂを含む幾つかの制御レジスタを有する（図３）。分岐は、部分的に、目標アドレスレジスタ４６ａの１つ又はもう一方の内容に基づいて処理される。特定の目標アドレスレジスタには、当該分岐の準備中に到来する分岐命令に先つ任意の時点において、分岐準備（PT）命令の使用によ

り、目標アドレスの書き込みが可能である。更に十分に検討されるように、分岐の事前準備に目標アドレスレジスタ４６ａを使用すれば、当該分岐のペナルティが軽減される。

【００４１】（ＳＲ）４６ｂは、実行の現行スレッドによって実行される命令の動作を制御するためのフィールドを含む制御レジスタである。ＳＲ４６ｂのレイアウトを図３に示す。「ｒ」フィールド（ビット位置０、２－３、１０－１１、２４－２５、２９、３２－６３）は予約済みビットを示す。簡潔に述べれば、本発明に関係するＳＲ４６ｂのフィールドは次のように作用する。

【００４２】１ビットフィールドＳ、Ｑ、Ｍ（各々のビット位置は１、８、９）は本発明の理解に関連しない特定の算術演算に際してモードＡ命令によるモードＢ命令のエミュレーションに用いられる。これらのビット位置は、モードＡ命令セットアーキテクチャによりモードＢ命令をエミュレートする際に用いられる、モードＢ命令セットアーキテクチャ環境からマップされた状態である。

【００４３】１ビットフィールドＦＲ、ＳＺ、ＰＲ（各々ビット位置は１４、１３、１２）はモードＢ浮動小数点命令の追加演算コード認定を供給するために用いられる。

【００４４】モードＢ命令セットアーキテクチャは、なかんずく、符号なし加算演算から得られる桁上げビットを維持するためにも１ビットＴレジスタを使用する。モードＢのＴレジスタは、上述したように、汎用レジスタR_{25}のＬＳＢにマップされる。他のマッピングを以下に説明する。ただし、特定のマッピングが、エミュレートされる特定の命令セットアーキテクチャおよび当該エミュレーションを実施する命令セットアーキテクチャに依存することは当該技術分野の当業者によって理解されるはずである。

【００４５】一旦、ＤＥＣ４４によって命令が復号されると、ＰＰＣ４８は、例えばＬＳＵ２８、及び／又は、ＩＭＵ３２のような残りのパイプステージを介してそれらの実行を監視する。ＰＰＣ４８の主要機能は、命令が正しくかつ円滑に実行されること、及び、（１）全てのソースオペランドが、必要に応じて（ＩＭＵ３２乗算累積内部伝送するために）、準備が整っているか、又は、準備を整えることが可能になるまで、命令がデコード復号ステージ段階に保持されること、（２）命令ならびに全ての内部／外部事象によって課せられる全ての同期化および直列化必要条件が維持されていること、（３）全てのデータオペランド／一時的結果が正しく伝送されていること、を保証することである。

【００４６】ＰＰＣ４８の制御論理を簡素化するために、モードＡ命令セット実行に関する幾つかの観察び仮定が行われる。これらの仮定の１つは、どのＩＭＵ命令も、決定論的に、例外を起こすこと、及び、全ての流れがパイプステージを通過するようにすることは不可能であるということである。この仮定は、ＰＰＣ４８が、ＩＭＵ３２を、入力オペランドの源および出力結果の行き先を知る必要のない複雑なデータ演算エンジンとして見ることを可能にする。

【００４７】ＰＰＣ４８の他の主要機能は、例えば命令例外、外部割込み、リセット等のような非順次的事象を処理することである。正常な実行条件の下において、ＰＰＣ４８のこの部分は常にアイドル状態にあり、事象が発生するとき、目をさます。ＰＰＣ４８は、外部の割込みコントローラ（図示せず）から外部割り込み／リセット信号、及び、プロセッサエレメント１２の多くの部品から内部例外を受け取る。どちらの場合にも、ＰＰＣ４８はパイプラインを清掃し、コア状態および分岐をセーブするためにＢＲ４６に、また、分岐を適切なハンドラに通知する。複数の例外および割込みが同時に発生する時には、ＰＰＣ４８の例外割込み調停論理４８ａは、構造的に規定された優先順位に従ってそれらの間で調停する。

【００４８】レジスタR_0 －R_{63}を含む上述した汎用レジスタはＩＦＵ２６のレジスタファイル（ＯＦ）５０に見い出される。汎用レジスタ各々の広さないし大きさは６４ビットである。ＯＦ５０の制御はＰＰＣ４８による。また、汎用レジスタR_{63}は６４ビット固定（「０」）である。

【００４９】ＤＥＣ４４のモードＢトランスレータ４４ａはモードＢ命令をモードＡ命令のシーケンスに変換することに責任がある。変換された命令は、次に、復号するためにＤＥＣのモードＡデコーダ４４ｂに伝達される。モードＢ変換に関して、ＤＥＣはＦＥ４２の命令バッファ４２ａの底から１６ビットを参照し、モードＢ命令をエミュレートするために１サイクル当たり１つのモードＡ命令を発出する。モードＡ命令は、ＦＥ４２のマルチプレクサ４３へ、次にモードＡデコーダ４４ｂまでルートバックされる。変換状態は、モードＢエミュレートシーケンスの生成を制御するために、ＤＥＣ４４内に維持される。全てのエミュレート命令が生成されるとき、ＤＥＣ４４は、次のモードＢ命令にシフトすることをＦＥ４２に通知する。この場合のシフト位置は命令バッファ４２ａの最上ビットから１６ビット又はバッファの下から１６ビットである。

【００５０】図４はＦＥ４２及びＤＥＣ４４を更に詳細に示す。図４に示すように、命令バッファ（ＩＢ）４２ａはＩＣＣからフェッチされた命令を受け取る。命令は、ＩＢ４２ａから引っ張られ、動作のモード（即ち、モードＢ命令が使用中であって、モードＡ命令によってエミュレート中であるか、又は、モードＡ命令のみが使用中であるか）に応じて、ＤＥＣ４４のモードＢトランスレータ４４ａ及びモードＡプリデコーダ４４ｃに供給される。予め複号済みのモードＡ命令（モードＡで作動

中であれば）、又は、モードBトランスレータからのモードA命令（モードBで作動中であれば）は、モードAデコーダ44bへ供給するためにマルチプレクサ43によって選定される。モードAプリデコーダは、32ビット命令信号に幾らかのプリデコード信号を加えた信号を生成する。モードBトランスレータは、モードA32ビット命令信号にデコード復号信号を加えた信号も生成し、この信号は、モードA命令が供給済みであれば、モードAプリデコーダによって生成されるプリデコード信号をエミュレートする。

【0051】FE42は、何の実行モードが存在するか、即ち、実行されているのはモードA命令であるか、又は、実行するためにモードA命令に変換されているのはモードB命令であるかを表示するために設定されるモードラッチ42bを含む。モードラッチ42bはマルチプレクサ43を制御する。理解されるように、本発明によれば、命令実行のモードは、分岐命令の目標アドレスの最下位ビット（LSB）によって決定される。モードA環境において作動中である時には、「0」にセットされた目標命令のアドレスのLSBを持ったモードAの無条件分岐命令（BLINK）を用いてモードBへの切り替えが実施される。モードBからモードAへの切り替えは、「1」にセットされたLSBを持つ目標アドレスを用いて、数個のモードB分岐命令によって開始される。

【0052】「delay slot present」（遅延スロット在り）ラッチ42cはFE42内に存在する。DSP42cは、変換され、エミュレートされ、分岐が実施可能になる以前に実行されるべき遅延スロット命令が変換中のモードB分岐命令に後続することを表示するように、DEC44のモードBトランスレータ44aからの信号によって設定される。遅延スロット命令が変換のためにモードBトランスレータ44aに送られるとき、DSP42eは、FE42によってリセットされる。

【0053】図4は、モードBトランスレータ44a、モードAデコーダ44b、及び、モードAプリデコーダ44cを含むDEC44を示す。モードB命令は、FE42から発出され、緩衝され、モードBトランスレータ44a、即ち、各モードB命令に対して1つ又は複数のモードA命令を生成する状態マシン実現回路に供給される。トランスレータ44aによって生成されたモードA命令はマルチプレックサ回路43を通過し、緩衝された後で、モードAデコーダ44bに供給される。次に、復号済み命令（即ち、オペランド、命令信号、等々）は実行ユニットに伝達される。

【0054】プロセッサエレメントの演算性能は分岐の効率に高度に依存する。従って、制御流れ機構は、低ペナルティ分岐をサポートするように設計されている。これは、CPUに当該分岐目標について通知する準備目標（PT）命令を、流れに対する制御、おそらく条件つきで当該分岐目標に対する制御を起こさせる分岐命令から分離することによって、本発明により達成される。この技法は、ハードウェアが、多くのサイクル数だけ先立って、分岐目標について通知されることを可能にし、分岐が実施される場合には、ハードウェアが命令の現行シーケンスから目標シーケンスへの円滑な遷移を準備することを可能にする。また、この段階において、分岐は比較演算の包括的セットを符号化するのに十分な空間を持つので、この配置構成は、分岐命令における一層多くの融通性を可能にする。これらは、1つの単一命令内に比較と分岐両方のオペレーションを含むので、折畳み比較分岐と呼ばれる。

【0055】モードB命令セットアーキテクチャに用いられるレジスタは一般に広さが32ビットであり、その個数（例えば16）はモードA命令セットアーキテクチャに使用されるレジスタの個数（64個、広さは各64ビット）よりも少ないものであることもあり得る。従って、モードB命令実行用汎用レジスタは、OF50のモードA汎用レジスタ16個の下から32ビットにマップされる。更に、上述したように、符号付きエキステンションが用いられる。即ち、オペランド又はモードB命令の他の表現式がOF50の汎用レジスタに書き込まれる時、上側ビット位置（32－63）にコピーされた最上位ビット（ビット位置31）を伴った下側ビット（ビット位置0－31）に書き込まれる。更に、モードB命令セットに用いられる状態レジスタの状態は、モードAアーキテクチャの特定レジスタビットにマップされる。

【0056】図5に示すマッピングの例は、早期開発されたモードB命令セットアーキテクチャの状態、及び、それがマップされた、モードAアーキテクチャ状態を示す。特定の命令セットに関して、本技術分野における当業者は、マッピングが利用可能な資源に依存することを認識するはずである。従って、ここに示す特定のマッピングは、当該マッピングに関連する命令セットアーキテクチャに依存する事例にすぎない。図5は、モードA状態（右端欄）へのモードB状態（左端欄）のマッピングを示す。例えば、モードBアーキテクチャのプログラムカウンタ状態は、モードAアーキテクチャのプログラムカウンタの低位ビット位置にマッピングされる。

【0057】レジスタマッピングに加えて、種々フラグの状態もマップされる。図5に示すように、1ビットフラグは、モードAアーキテクチャの一般レジスタの1つ又はもう一方の特定ビット位置にマップされる。従って、例えば、モードA　T、S、M、Q状態／フラグは、汎用レジスタR_{25}（ビット位置0）、及び、SR46b（フィールドS、M、Q）にそれぞれマップされる。

【0058】広さ32ビットのモードA命令は4バイト境界上に記憶され、モードB命令は4バイト又は2バイト境界のどちらかに記憶される。従って、少なくとも2

ビット（ＬＳＢおよびＬＳＢ＋１）はアドレッシング用として未使用であり、オペレーションのモードを識別するために利用可能である。モードＡとモードＢ命令実行の間のスイッチングは、分岐の目標アドレスの２つのＬＳＢを検出する分岐命令を用いて達成される。モードＡ命令の実行に際して、モードＡオペレーションからモードＢオペレーションへスイッチすることは無条件分岐アドレス（ＢＬＩＮＫ）のみが可能である。従って、オペレーションのモードは、モードＡおよびＢ命令セットアーキテクチャに用いられるジャンプ命令の目標アドレスのＬＳＢを用いて変更可能である。このビット位置における「０」はモードＢ目標命令を表示し、「１」はモードＡ目標命令を表示する。ＬＳＢは、モード表示にのみ用いられ、実際の目標アドレスに影響しない。

【００５９】初期のモードＢ命令セットアーキテクチャは、分岐（分岐）オペレーションに関して生じるペナルティを軽減するために遅延スロット機構を利用した。遅延スロットは分岐命令に密着して後続する命令であり、当該分岐がプログラム流において遷移を引き起こす（又は、引き起こさない）ことができる以前に実行される。既に示したように、当該分岐に充分先立って目標アドレスレジスタに当該分岐の目標アドレスをロードするために、一層円滑な遷移をＰＴ命令によって実施可能である。ただし、遅延スロット付きモードＢ分岐命令のエミュレーションは当該遅延スロットについて解明しなければならない。従って、遅延スロット付きモードＢ分岐命令に遭遇した時には、モードＡコードシーケンスは当該分岐を実施するが、目標命令は、当該分岐命令（即ち、遅延スロット命令）に後続するモードＢ命令がエミュレートされて完了するまで実行されない。

【００６０】図６は本発明の一態様のモードＡ及びモードＢ環境両方におけるＰＴ命令の使用を含む用法、モードＡスレッド５８から更にコンパクトなモードＢスレッドへのスイッチング（ステップ６４－８４）、及び、モードＡスレッドへの復帰を示す。

【００６１】本発明の理解は、分岐命令のオペレーションについての記述から最もよく実現可能である。

【００６２】モードＡからモードＡへの分岐：図４および６を参照することとし、モードＡ命令ストリーム５８（図６）が実行中である間に、ＢＬＩＮＫ命令を用いて無条件分岐が別のモードＡ命令ストリームに対して（即ち、モード切り替えなしに）実施されるものと仮定する。ＰＴ命令は、復号するためにＩＢ４２ａからＢＬＩＮＫ命令がプルされる幾らか以前に、８個の目標アドレスレジスタ４６ａの１つに、そこに分岐が実施されるべき目標アドレスをロードする（ステップ６０）。その後で、ＢＬＩＮＫ命令は、ＩＢ４２ａの最上部に到達し、部分的復号のために、モードＡプリデコーダ４４ｃに送られ、次に、マルチプレクサ４３を介してＤＥＣ４４のモードＡデコーダ４４ｂまで送られる（ステップ６２）。ＤＥＣ４４は、目標命令のアドレスを含む目標アドレスレジスタ４６ａの識別表示を有する復号済み情報をＢＲ４６に送る。

【００６３】ステップ６６において、ＢＲ４６は、識別された目標アドレスレジスタ４６ａから目標アドレスを読み取り、それを、分岐コマンド信号と共にＦＥ４２に送る。続いて、ＢＲは、分岐命令に後続する実行パイプライン内の全ての命令を無効化する。

【００６４】一方、ＦＥ４２は、ステップ６８において、ＩＣＵ２７から目標命令をフェッチするために、ＢＲ４６から受け取った目標アドレスを用いてフェッチリクエストをＩＣＣ４０へ発出する（図２）。ステップ７０において、ＦＥ４２は、目標アドレスのＬＳＢをチェックする。ＬＳＢが「０」であれば、ＦＥは、目標命令がモードＢ命令であることを知るはずである。ただし、この場合、目標命令はモードＡ命令であるので、ＬＳＢは「１」であり、モード変化は起こらない。ほぼ同時に、目標命令とそれに後続する命令の命令ストリームを受取る準備のために、ＩＢ４２ａの内容は無効化される。目標命令は、ＩＣＣ４０から受け取られると、ＩＢ４２ａ内に置かれ、復号するために、そこからＤＥＣ４４に送られ、ステップ７２において、オペレーションは継続する。

【００６５】モード切り替え：モードＡからモードＢへの分岐：ここで、モードＡ命令シーケンス内において、モードＢシーケンスの更にコンパクトなコードへの切り替え（スイッチ）が行われるものと仮定する。この点で、目標アドレスのＬＳＢの使用が役立つことになる。初めに、目標命令がモードＢ命令であることを表示するために「０」に設定されたＬＳＢを有する目標アドレスを、ＰＴ命令が、目標アドレスレジスタ４６ａにロードすることをステップ６０が確認することを除けば、ステップ６０－６８は、上述の場合と同じである。次に、モードＡ実行からモードＢ実行へのスイッチ用に用いられるＢＬＩＮＫ分岐命令がＤＥＣ４４に送られて、復号される（ステップ６２）。ＢＬＩＮＫ命令を復号した後で、ＤＥＣ４４は、分岐用に使用するために、目標アドレスレジスタ４６ａの識別をＢＲ４６に送る。一方、ＢＲは、識別済み目標アドレスレジスタ４６ａの内容を読み取り、分岐コマンド信号と共にそれをＦＥ４２に送り（ステップ６６）、分岐命令に後続する実行パイプライン内のあらゆる命令を無効化する。ＦＥ４２は、目標アドレスを用いてフェッチリクエストをＩＣＣ４０に送り、その代わりに、目標命令を受け取る（ステップ６８）。更に、ステップ７０において、ＦＥは、目標アドレスの下位ビット（即ち、ＬＳＢ）が「０」であることを検出し、モードＢオペレーションを表示するようにモードラッチ４２ｂを適宜設定することによって、その内部モード状態をモードＡからモードＢへ変える（ステップ７６）。モードラッチ４２ｂの出力は、モードＢトラ

ンスレータ４４ａからモードＡデコーダ４４ｂへ命令を通信するようにマルチプレクサ４３を制御する。

【００６６】ここで、スイッチは完了する。次に、命令はモードＢトランスレータへ送られ（ステップ７８）、ここで、これらの命令はモードＢ命令をエミュレートするモードＡ命令に変換される。

【００６７】モードＢからモードＢへの分岐：モードＢにおいて作動中の期間における分岐は基本的に上述した場合と同じである。モードＢ分岐命令は、目標レジスタ４６ａに目標命令のアドレスをロードするＰＴ命令を含むモードＡ命令のシーケンスに変換され、分岐を実行するために、モードＡ分岐命令（例えばＢＬＩＮＫ分岐命令）がこれに続く。当該分岐が実施可能である以前に実行されなければならない分岐命令に後続する遅延スロット命令をモードＢ分岐命令が表示するならば例外である。遅延スロット命令がモードＢ分岐命令に後続しないならば、目標命令のアドレスを提供するためにＰＴ命令によって先行された後、モードＡ分岐に関して既に概説したステップが実施される。

【００６８】しかし、遅延スロット命令が存在するならば、モードＢトランスレータ４４ａは、分岐命令を復号し、それが遅延スロット命令の存在を表示することを注記することによって、ＦＥに対して遅延スロットが存在することを表示するＦＥにおけるラッチ４２ｃをセットするためにＤＳ．ｄ信号をＦＥ４２に表明する。ＢＲ４６が分岐目標アドレスをＦＥ４２に送る時には、ＦＥ４２は、遅延スロット命令を除くＩＢ４２ａの全ての内容を無効化する。ＦＥは、目標命令をフェッチすることをＩＣＣ４０に要求し、遅延スロット命令が未だＤＥＣ４４に転送されていなければ、それを受け取ると、それを遅延スロット命令の背後に置く。また、ＦＥ４２は、分岐目標アドレスのＬＳＢを調べる。ＬＳＢが「０」であるならば、モードビット４２ｂは変更されない状態のままである。

【００６９】遅延スロット命令は、モードＢトランスレータに供給され、それをエミュレートするモードＡ命令を生成するために変換され、次に、ＦＥ４２はＤＳＰ４２ｃを「０」にリセットする。遅延スロット命令のエミュレーションが完了すると、分岐目標命令はモードＢトランスレータへ供給される。

【００７０】モード切り替え：モードＢからモードＡへの分岐：再び、モードＢ命令が実行中であっても、実施される初期ステップは基本的に上述したものと同じである。モードＢ分岐命令は、モードＢトランスレータによって、目標アドレスレジスタにモードＡ目標命令の目標アドレス（ＬＳＢは「１」に設定）をロードするためのＰＴ命令を有するモードＡ命令シーケンスを生成するように変換される。また、モードＢ分岐命令がそれに後続する遅延スロット命令を持つならば、モードＢトランスレータはＤＳ．ｄ信号をＦＥに発出し、遅延スロット命令が存在することを表示するためにＦＥのＤＳＰラッチ４２ｃを設定する。ＢＲは、目標がモードＡ命令であることを表示するために、そのＬＳＢが「１」に設定された目標アドレスの内容を読み取り、それをＦＥ４２に送る。次に、ＢＲ４６は、パイプライン内に偶然存在するならば遅延スロット命令のエミュレーションを除き、分岐命令に後続するパイプライン内の全ての命令を無効化する。

【００７１】目標アドレスを受け取ると、ＦＥ４２は、目標アドレスを用いてＩＣＣ４０にフェッチリクエストを発出し、遅延スロット命令を除くＩＢ４２ａの内容を無効化する。遅延スロットが変換された後で、モードＡオペレーションを表示するようにモードラッチを設定することによりＦＥ４２はそのモード状態を変える。ここにおいて、目標命令を含むＩＢ４２ａからの更なる全ての命令はマルチプレクサ４３によってモードＡプリデコーダ４４ｃに経路指定される。

【００７２】

【発明の効果】本発明によれば、以下の優れた効果が得られる。

【００７３】(1).１６ビットコード及び３２ビットコードの両方を実行する能力は、プロセッサが普通のタスクのためにコンパクトな１６ビットコードを使用することを可能にする。

【００７４】(2).両メモリスペースの節減および当該節減を伴った他の利点（例えば、更に小さいメモリ、節減された電力消費等）を可能にする。更に複雑なタスクが必要とされるときは、３２ビットコードが使用できる。

【００７５】(3).初期に設計された１６ビット命令セットアーキテクチャを実行する能力は、当該初期設計においてなされた投資の保持を可能にする互換性を提供する。

【００７６】(4).ＰＴ命令は、分岐に関する事前通知を提供することにより、分岐命令の遂行におけるより多くの融通性を可能にする。

【００７７】以下に本発明の付録Ａ，Ｂを示す。

【００７８】付録Ａ

モードＢデータ転送命令エミュレーションシーケンス

【００７９】

【表１】

モードB命令	モードA命令シーケンス	入	出
MOV #imm,Rn 1110 nnnn siii iiii	1 movi #imm,Rn		
MOV.W @(disp,PC),Rn 1001 nnnn dddd dddd	1 mova.w disp,R32 2 ld.w R32,#0,Rn		
MOV.L @(disp,PC),Rn 1101 nnnn dddd dddd	mova.l disp,R32 2 ld.l R32,#0,Rn		
MOV Rm,Rn 0110 nnnn mmmm 0011	1 addi Rm,#0,Rn		
MOV.B Rm,@Rn 0010 nnnn mmmm 0000	1 st.b Rn,#0,Rm		
MOV.W Rm,@Rn 0010 nnnn mmmm 0001	1 st.w Rn,#0,Rm		
MOV.L Rm,@Rn 0010 nnnn mmmm 0010	1 st.l Rn,#0,Rm		
MOV.B @Rm,Rn 0110 nnnn mmmm 0000	1 ld.b Rm,#0,Rn		
MOV.W @Rm,Rn 0110 nnnn mmmm 0001	1 ld.w rm,#0,Rn		
MOV.L @Rm,Rn 0110 nnnn mmmm 0010	1 ld.l rm,#0,Rn		
MOV.B Rm,@-Rn 0010 nnnn mmmm 0100	1 st.b Rn,#-1,Rm 2 addi.l Rn,#-1,Rn		
MOV.W Rm,@-Rn 0010 nnnn mmmm 0101	1 st.w Rn,#-1,Rm 2 addi.l Rn,#-2,Rn		
MOV.L Rm,@-Rn 0010 nnnn mmmm 0110	1 strd.w Rn,#-1,Rm 2 addi.l Rn,#-4,Rn		
MOV.B @Rm+,Rn 0110 nnnn mmmm 0100	1 ld.b Rm,#0,Rn *2 if (m!=n)* addi.l Rm,#1,Rm		
MOV.W @Rm+,Rn 0110 nnnn mmmm 0101	1 ld.w Rm,#0,Rn *2 if (m!=n)* addi.l Rm,#2,Rm		
MOV.L @Rm+,Rn 0110 nnnn mmmm 0110	1 ld.l Rm,#0,Rn *2 if (m!=n)* addi.l Rm,#4,Rm		
MOV.B R0,@(disp,Rm) 1000 0000 mmmm dddd	1 st.b Rn,disp,R0		
MOV.W R0,@(disp,Rm) 1000 0001 mmmm dddd	1 st.w Rn,disp,R0		
MOV.L Rm,@(disp,Rn) 0001 nnnn mmmm dddd	1 st.l Rn,disp,Rm		
MOV.B @(disp,Rm),R0 1000 0100 mmmm dddd	1 ld.b Rm,disp,R0		
MOV.W @(disp,Rm),R0 1000 0101 mmmm dddd	1 ld.w Rm,disp,R0		

【0080】 * *【表2】

MOV.L @(disp,Rm),Rn 0101 nnnn mmmm dddd	1 ld.l Rm,disp,Rn		
MOV.B Rm,@(R0,Rn) 0000 nnnn mmmm 0100	1 stx.b Rn,R0,Rm		
MOV.W Rm,@(R0,Rn) 0000 nnnn mmmm 0101	1 stx.w Rn,R0,Rm		
MOV.L Rm,@(R0,Rn) 0000 nnnn mmmm 0110	1 stx.l Rn,R0,Rm		
MOV.B @(R0,Rm),Rn 0000 nnnn mmmm 1100	1 ldx.b Rm,R0,Rn		
MOV.W @(R0,Rm),Rn 0000 nnnn mmmm 1101	1 ldx.w Rm,R0,Rn		
MOV.L @(R0,Rm),Rn 0000 nnnn mmmm 1110	1 ldx.l Rm,R0,Rn		
MOV.B R0,@(disp,GBR) 1100 0000 dddd dddd	1 stx.b R27,disp,R0		
MOV.W R0,@(disp,GBR) 1100 0001 dddd dddd	1 stx.w R27,disp,R0		
MOV.L R0,@(disp,GBR) 1100 0010 dddd dddd	1 stx.l R27,disp,R0		
MOV.B @(disp,GBR),R0 1100 0100 dddd dddd	1 ld.b R27,disp,R0		
MOV.W @(disp,GBR),R0 1100 0101 dddd dddd	1 ld.w R27,disp,R0		
MOV.L @(disp,GBR),R0 1100 0110 dddd dddd	1 ld.l R27,disp,R0		
MOVA @(disp,PC),R0 1100 0111 dddd dddd	1 mova.l disp,R0		
MOVT Rn 0000 nnnn 0010 1001	1 andi R25,#1,Rn		
SWAP.B Rm,Rn 0110 nnnn mmmm 1000	1 bytarev Rm,R32 2 shlri Rm,#16,Rn 3 mextr6 R32,Rn,Rn	Rm	
SWAP.W Rm,Rn 0110 nnnn mmmm 1001	1 mperm.w Rm,#1,R32 2 addi.l R32,#0,Rn		
XTRCT Rm,Rn 0010 nnnn mmmm 1101	1 shlli.l Rm,#16,R32 2 shlri.l Rn,#16,Rn 3 or Rn,R32,Rn		

算術命令エミュレーションシーケンス

モードB命令	モードA命令シーケンス	入	出
ADD Rm,Rn 0011 nnnn mmmm 1100	1 add.l Rm,Rn,Rn		
ADD #imm,Rn 0111 nnnn siii iiii	1 addi.l Rn,#imm,Rn		
ADDC Rm,Rn 0011 nnnn mmmm 1110	1 addz.l Rm,R63,R32 2 addz.l Rn,R63,Rn		

【0081】 * *【表3】

25 26

	3 add Rn,R32,Rn 4 add Rn,R25,Rn 5 shlri Rn,#32,R25 6 addl.l Rn,#0,Rn		
ADDV Rm,Rn 0011 nnnn yccc 1111	1 add Rm,Rn,R32 2 addJ Rm,Rn,Rn 3 cmpne Rn,R32,R25	Rm Rn	
CMP/EQ #imm,R0 1000 1000 siii iii	1 movi #imm,R32 2 cmpeq R0,R32,R25	R0	
CMP/EQ Rm,Rn 0011 nnnn mmmm 0000	1 cmpeq Rn,Rm,R25	Rm Rn	
CMP/HS Rm,Rn 0011 nnnn mmmm 0010	1 cmpgeu Rn,Rm,R25	Rm Rn	
CMP/GE Rm,Rn 0011 nnnn mmmm 0011	1 cmpge Rn,Rm,R25	Rm Rn	
CMP/HI Rm,Rn 0011 nnnn mmmm 0110	1 cmpgtu Rn,Rm,R25	Rm Rn	
CMP/GT Rm,Rn 0011 nnnn mmmm 0111	1 cmpgt Rn,Rm,R25	Rm Rn	
CMP/PZ Rn 0100 nnnn 0001 0001	1 cmpge Rn,R63,R25	Rn	
CMP/PL Rn 0100 nnnn 0001 0101	1 cmpgt Rn,R63,R25	Rn	
CMP/STR Rm,Rn 0010 nnnn mmmm 1100	1 mcmpeq.b Rm,Rn,R32 2 addz.l R32,R63,R32 3 cmpgtu R32,R63,R25		
DIV0S Rn,Rm 0010 nnnn mmmm 0111	1 xor Rn,Rm,R32 2 ori Rn,#0,R33 3 ori Rm,#0,R33 4 shlrl.l R32,#31,R32 5 xori R32,#1,R25 *Q.d = PPF_EX2[63]* 6 nop *M.d = PPF_EX2[63]* *#1.5 dec_br_sr_update = 1*		
DIV0U 0000 0000 0001 1001	1 movi #0,R25 *Q.d = 0* *M.d = 0* *#1.5 dec_br_sr_update = 1*		
DIV1 Rm,Rn 0011 nnnn mmmm 0100	1 addz.l Rm,R63,R32 2 ori Rn,#0,R33 *oldQ.d = Q.q* 3 shll.l Rn,#1,Rn 4 addz.l Rn,R25,Rn *5 If (oldQ.q == M.q)* sub Rn,R32,Rn *else* add Rn,R32,Rn *Q.d = PPF_EX2[63]* 6 addl.l Rn,#0,Rn *oldQ.d = Q.q ^ M.q*		

【0082】 * *【表4】

27 28

	7 nop 8 nop *Q.d = oldQ.q ^ PPF_EX2[32]* 9 *If (Q.q == 1)* movi #1,R25 else movi #0,R25 *#1.5 dec_br_sr_update = 1*		
DT Rn 0100 nnnn 0001 0000	1 addl.l Rn,#-1,Rn 2 cmpeq Rn,R63,R25		
DMULS.L Rm,Rn 0011 nnnn mmmm 1101	1 muls.l Rm,Rn,R24		
DMULU.L Rm,Rn 0011 nnnn mmmm 0101	1 mulu.l Rm,Rn,R24		
EXTS.B Rm,Rn 0110 nnnn mmmm 1110	1 shll Rm,#56,Rn 2 shari Rn,#56,Rn		
EXTS.W Rm,Rn 0110 nnnn mmmm 1111	1 shll Rm,#48,Rn 2 shari Rn,#48,Rn		
EXTU.B Rm,Rn 0110 nnnn mmmm 1100	1 andi Rm,#255,Rn		
EXTU.W Rm,Rn 0110 nnnn mmmm 1111	1 shll Rm,#48,Rn 2 shlri Rn,#48,Rn		
MUL.L Rm,Rn 0000 nnnn mmmm 0111	1 mulu.l Rm,Rn,R32 2 shlri R24,#32,R24 3 mshflo.l R32,R24,R24		
MULS.W Rm,Rn 0010 nnnn mmmm 1111	1 mmulfo.wl Rm,Rn,R32 2 shlri R24,#32,R24 3 mshflo.l R32,R24,R24		
MULU.W Rm,Rn 0010 nnnn mmmm 1110	1 shll Rm,#48,R32 2 shll Rn,#48,R33 3 shlri R32,#48,R32 4 shlri R33,#48,R33 5 mulu.l R32,R33,R32 6 shlri R24,#32,R24 7 mshflo.l R32,R24,R24		
NEG Rm,Rn 0110 nnnn mmmm 1011	1 sub.l R63,Rm,Rn		
NEGC Rm,Rn 0110 nnnn mmmm 1010	1 addz.l Rm,R63,R32 2 sub R63,R25,Rn 3 sub Rn,R32,Rn 4 shlri Rn,#32,R25 5 addl.l Rn,#0,Rn	Rm	
SUB Rm,Rn 0011 nnnn mmmm 1000	1 sub.l Rn,Rm,Rn		
SUBC Rm,Rn 0110 nnnn mmmm 1010	1 addz.l Rm,R63,R32 2 addz.l Rn,R63,Rn 3 sub Rn,R25,Rn 4 sub Rn,R32,Rn 5 shlri Rn,#32,R25 6 addl.l Rn,#0,Rn		

【００８３】付録Ｂ

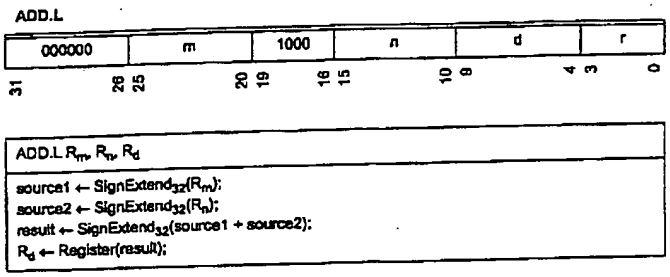
ＡＤＤ．Ｌ

説明：ＡＤＤ．Ｌ命令は、R_m の下位３２ビットをR_n の下位３２ビットに加算し、かつ符号拡張された３２ビットの結果をレジスタR_dに格納する。ビット$R_{m\,(32}$ ＊FOR 32) および$R_{n(32}$ FOR 32) は無視される。

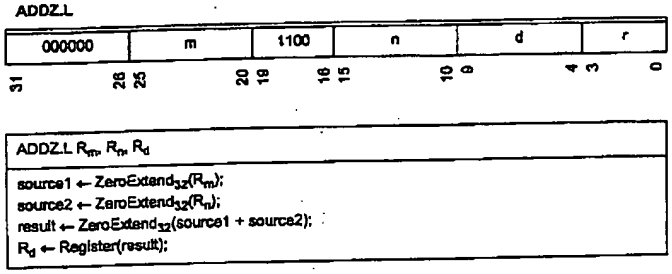
操作：

【００８４】

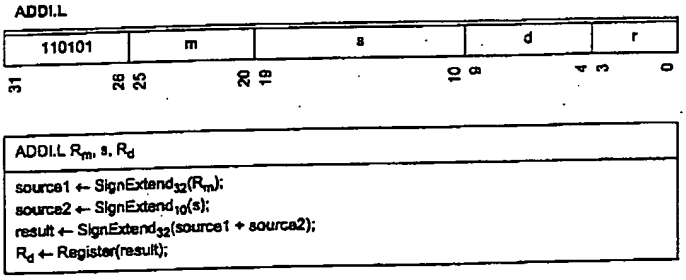
【表５】



【００８５】ＡＤＤＺ．Ｌ
説明：ＡＤＤＺ．Ｌ命令は、R_m の下位３２ビットをR_nの下位３２ビットに加算し、かつ零拡張された３２ビットの結果をレジスタR_dに格納する。ビットR_m (32 ＊FOR 32) およびR_n(32 FOR 32) は無視される。
操作：
【００８６】
【表６】

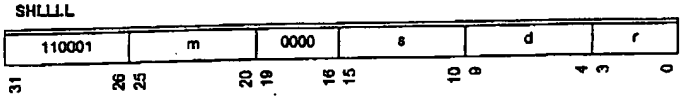


【００８７】ＡＤＤＩ．Ｌ
説明：ＡＤＤＩ．Ｌ命令は、R_m の下位３２ビットをインターネットミーディエイト（即値）ｓの符号拡張された値に加算し、かつ符号拡張された３２ビットの結果をレジスタR_dに格納する。ビットR_m (32 FOR 32) は無視される。
操作：
【００８８】
【表７】



【００８９】ＳＨＬＬＩ．Ｌ
説明：ＳＨＬＬＩ．Ｌ命令は、R_m の下位３２ビットをｓ(0FOR 5) だけ論理的に左シフトさせ、かつ符号拡張された３２ビットの結果をレジスタR_dに格納する。
★操作：
【００９０】
【表８】

SHLLI.L



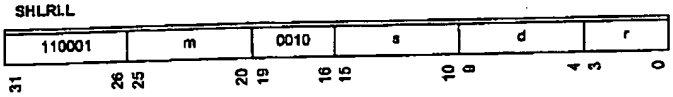
SHLLI.L R_m, s, R_d

source1 ← ZeroExtend$_{32}$(R_m);
source2 ← ZeroExtend$_5$(SignExtend$_6$(s));
result ← SignExtend$_{32}$(source1 << source2);
R_d ← Register(result);

【0091】SHLRI．L　　＊操作：

説明：SHLRI．L命令は、R_m の下位32ビットを　　【0092】
S(0FOR 5) だけ論理的に右シフトさせ、かつ符号拡張　　【表9】
された32ビットの結果をレジスタR_dに格納する。＊

SHLRI.L



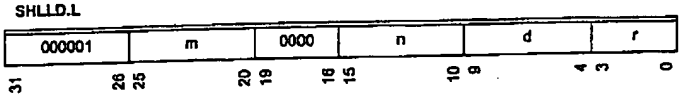
SHLRI.L R_m, s, R_d

source1 ← ZeroExtend$_{32}$(R_m);
source2 ← ZeroExtend$_5$(SignExtend$_6$(s));
result ← SignExtend$_{32}$(source1 >> source2);
R_d ← Register(result);

【0093】SHLLD．L　　※操作：

説明：SHLLD．L命令は、R_m の下位32ビットを　　【0094】
$R_{n(0\ FOR\ 5)}$ だけ論理的に左シフトさせ、かつ符号拡　　【表10】
張された32ビットの結果をレジスタR_dに格納する。※

SHLLD.L



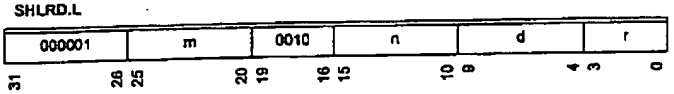
SHLLD.L R_m, R_n, R_d

source1 ← ZeroExtend$_{32}$(R_m);
source2 ← ZeroExtend$_5$(R_n);
result ← SignExtend$_{32}$(source1 << source2);
R_d ← Register(result);

【0095】SHLRD．L　　★操作：

説明：SHLRD．L命令は、R_m の下位32ビットを　　【0096】
$R_{n(0\ FOR\ 5)}$ だけ論理的に右シフトさせ、かつ符号拡　40　【表11】
張された32ビットの結果をレジスタR_dに格納する。★

SHLRD.L

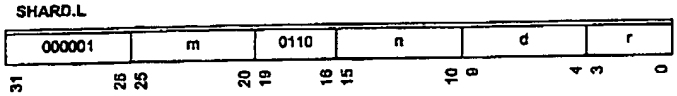


SHLRD.L R_m, R_n, R_d

source1 ← ZeroExtend$_{32}$(R_m);
source2 ← ZeroExtend$_5$(R_n);
result ← SignExtend$_{32}$(source1 >> source2);
R_d ← Register(result);

【0097】SHARD．L　　＊操作：
説明：SHARD．L命令は、R_m の下位32ビットを　　【0098】
$R_{n(0\ FOR\ 5)}$ だけ算術的に右シフトさせ、かつ符号拡　　【表12】
張された32ビットの結果をレジスタR_dに格納する。＊

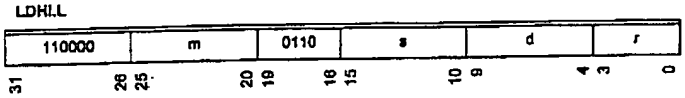
SHARD.L




```
SHARD.L Rm, Rn, Rd
source1 ← SignExtend32(Rm);
source2 ← ZeroExtend5(Rn);
result ← SignExtend32(source1 >> source2);
Rd ← Register(result);
```

【0099】LDHI．L　　※操作：
説明：不整合の、符号付けされたロング（長）ワードの　　【0100】
上位部分をメモリから汎用レジスタにロードする。　※　　【表13】

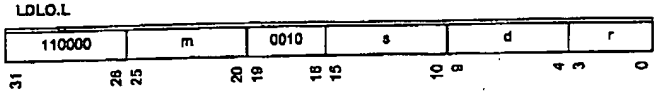
LDHI.L




```
LDHI.L Rm, s, Rd
base ← ZeroExtend64(Rm);
offset ← SignExtend6(s);
address ← base + offset;
count ← (address ∧ 0x3) + 1;
address ← ZeroExtend64(address ∧ (~ 0x3));
count8 ← count × 8;
shift ← ZeroExtend5((~ ((base + offset) ∧ 0x3)) × 8);
mem ← ZeroExtendcount8(MisalignedReadMemorycount8(address));
IF (IsLittleEndian())
   result ← SignExtend32(mem << shift);
ELSE
   result ← ZeroExtend32(mem);
Rd ← Register(result);
```

【0101】LDLO．L　　★操作：
説明：不整合の、符号付けされたロング（長）ワードの　　【0102】
下位部分をメモリから汎用レジスタにロードする。　★　　【表14】

LDLO.L



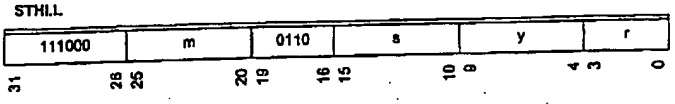

```
LDLO.L Rm, s, Rd
base ← ZeroExtend64(Rm);
offset ← SignExtend6(s);
address ← ZeroExtend64(base + offset);
count ← 4 - (address ∧ 0x3);
count8 ← count × 8;
shift ← (address ∧ 0x3) × 8;
mem ← ZeroExtendcount8(MisalignedReadMemorycount8(address));
IF (IsLittleEndian())
   result ← ZeroExtend32(mem);
ELSE
   result ← SignExtend32(mem << shift);
Rd ← Register(result);
```

【0103】STHI．L　　☆50☆説明：ロング（長）ワードの上位部分を汎用レジスタか

らメモリに不整合状態でストアする。

操作：

STHI.L




```
STHI.L Rm, s, Ry
base ← ZeroExtend64(Rm);
offset ← SignExtend6(s);
value ← ZeroExtend32(Ry);
address ← base + offset;
count ← (address ∧ 0x3) + 1;
address ← ZeroExtend64(address ∧ (~ 0x3));
IF (IsLittleEndian())
   start ← (4 - count) × 8;
ELSE
   start ← 0;
count8 ← count × 8;
MisalignedWriteMemory count8(address, value< start FOR count8 >);
```

【0105】STLO．L

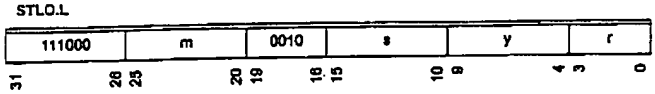
説明：ロング（長）ワードの下位部分を汎用レジスタからメモリに不整合状態でストアする。

※操作：

【0106】

【表16】

STLO.L




```
STLO.L Rm, s, Ry
base ← ZeroExtend64(Rm);
offset ← SignExtend6(s);
value ← ZeroExtend32(Ry);
address ← ZeroExtend64(base + offset);
count ← 4 - (address ∧ 0x3);
IF (IsLittleEndian())
   start ← 0;
ELSE
   start ← (4 - count) × 8;
count8 ← count × 8;
MisalignedWriteMemory count8(address, value< start FOR count8 >);
```

【図面の簡単な説明】

【図1】本発明を実現するように構成されたプロセッサエレメントを用いた処理システムを概略的に示すブロック図である。

【図2】図1に示すプロセッサエレメントの命令フェッチユニット（ＩＦＵ）を示すブロック図である。

【図3】図2に示す分岐ユニットに含まれる状態レジスタのレイアウトを示す図である。

【図4】図2に示すデコーダ（ＤＥＣ）を示すブロック図である。

【図5】1つの命令セットアーキテクチャから別の命令セットアーキテクチャへの状態マッピングを示す図である。

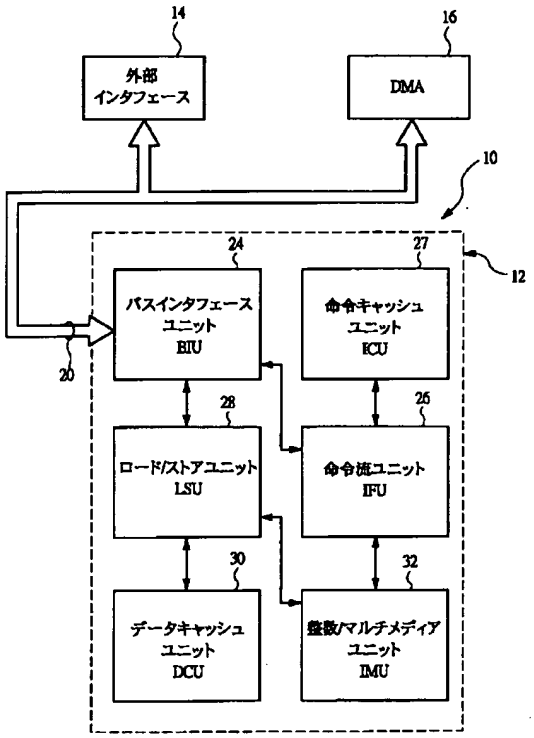
【図6】命令の流れを制御する本発明のアスペクトを示す★すフローチャートである。

【符号の説明】

１２　プロセッサエレメント
１４　直接メモリアクセスユニット
２０　システムバス
２４　ＢＩＵ：バスインタフェースユニット
２６　ＩＦＵ：命令流ユニット
２７　ＩＣＵ：命令キャッシュユニット
３０　ＤＣＵ：データキャッシュユニット
４０　ＩＣＣ：命令キャッシュコントロール
４２　ＦＥ：フェッチユニット
４４　ＤＥＣ：復号ユニット
４８　ＰＰＣ：パイプラインコントロール
５０　レジスタファイル

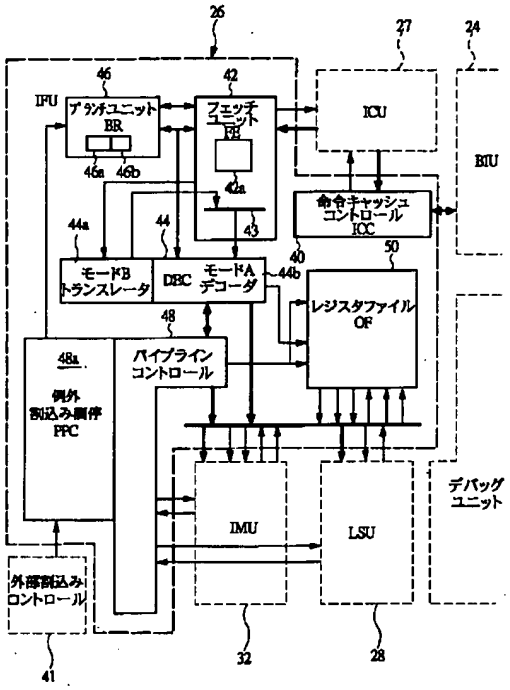
【図1】

図 1



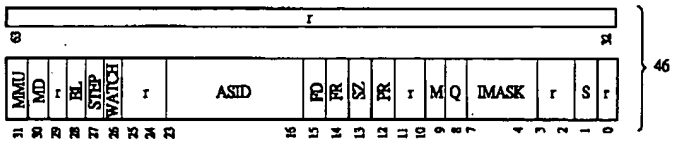
【図2】

図 2



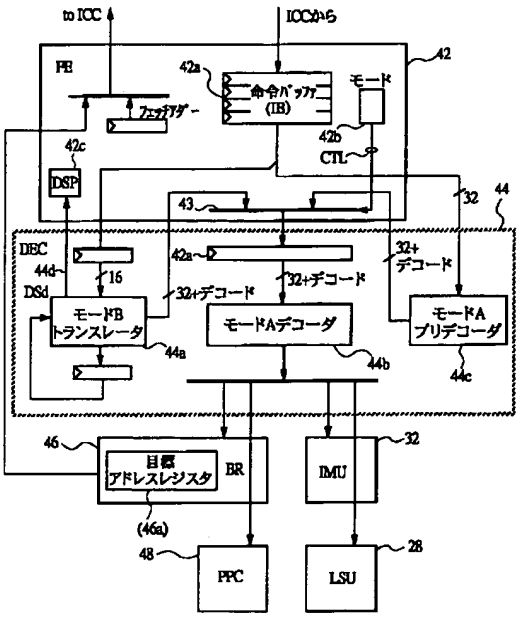
【図3】

図 3



【図4】

図 4



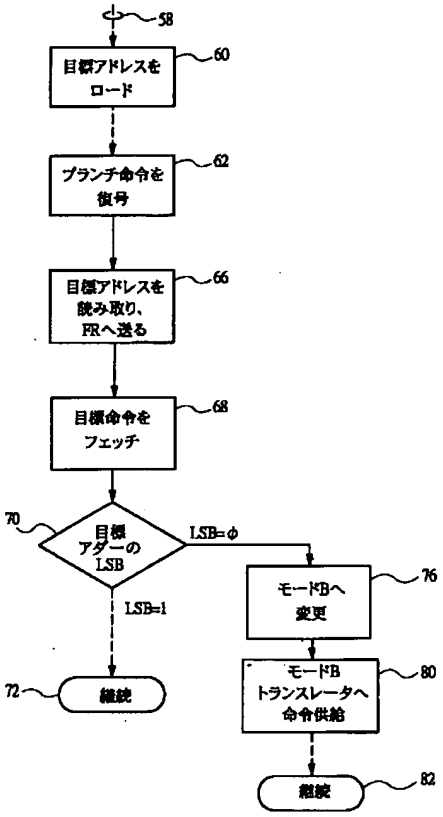
【図5】

図 5

モードB状態	記　述	モードA状態
PC	プログラムカウンタ	PCの下側32ビット
Ri、i= [0、15]	モードB汎用レジスタ	Riの下側32ビット、 i= [0,15]
PR	手順リンクレジスタ	R18の下側32ビット
GBR	大域ベースレジスタ	R27の下側32ビット
MACL	乗算累積ロー	R24の下側32ビット
MACH	乗算累積ハイ	R24の上側32ビット
T	条件コードフラグ	R25のビット0
S	乗算累積飽和フラグ	SR. S
M	ディバイドステップMフラグ	SR. M
Q	ディバイドステップQフラグ	SR. Q

【図6】

図 6



フロントページの続き

(72)発明者　マーク・デバージ
アメリカ合衆国、カリフォルニア州 95014、クペルティノ、スイート　371、スティーブンス・クリーク・ブルバード 5636

(72)発明者　セバスチャン・ハブルジ・ジースラー
アメリカ合衆国、カリフォルニア州 95118、サンノゼ、ジョセフ・レーン 5315

(72)発明者　カナド・ロイ
アメリカ合衆国、カリフォルニア州
95054、サンタクララ、ナンバー111、パーク・ビュー・ドライブ　610

(72)発明者　アンドリュー・クレイグ・スタルジス
イギリス国、バス　BA1　7RT、バス
フォード、パンプ・レーン　3

(72)発明者　プラセンジット・ビスワス
アメリカ合衆国、カリフォルニア州
95070、サラトガ、パンパス・コート
20167